An Adaptive Opposition-based Learning Selection: The Case for Jaya Algorithm

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.Doi Number

An Adaptive Opposition-based Learning Selection: The Case for Jaya Algorithm

Abdullah B. Nasser[1], Kamal Z. Zamli[1], Fadhl Hujainah[2], Waheed Ali H. M. Ghanem[3], Abdul-Malik H. Y. Saad[4] and Nayef Abdulwahab Mohammed Alduais[5]

[1] Faculty of Computing, College of Computing and Applied Sciences, Universiti Malaysia Pahang, 26600, Pekan, Pahang, Malaysia
[2] Computer Science and Engineering Department, Chalmers and University of Gothenburg, 41296 Gothenburg, Sweden
[3] Faculty of Ocean Engineering Technology and Informatics, Universiti Malaysia Terengganu, Kuala Terengganu, 21030, Malaysia
[4] Division of Electronic and Computer Engineering, School of Electrical Engineering, Faculty of Engineering, Universiti Teknologi Malaysia, 81310 JB, Johor, Malaysia.
[5] Faculty of Computer Science and Information Technology (FSKTM), Universiti Tun Hussein Onn Malaysia (UTHM), 86400, Parit Raja, Batu Pahat, Johor Malaysia

Corresponding author: Abdullah B. Nasser[1] (e-mail: abdullahnasser@ump.edu.my).

This work was supported by Universiti Malaysia Pahang under Grant FRGS/1/2019/ICT02/UMP/02/13 (RDU1901209).

ABSTRACT Over the years, opposition-based Learning (OBL) technique has been proven to effectively enhance the convergence of meta-heuristic algorithms. The fact that OBL is able to give alternative candidate solutions in one or more opposite directions ensures good exploration and exploitation of the search space. In the last decade, many OBL techniques have been established in the literature including the Standard-OBL, General-OBL, Quasi Reflection-OBL, Centre-OBL and Optimal-OBL. Although proven useful, much existing adoption of OBL into meta-heuristic algorithms has been based on a single technique. If the search space contains many peaks with potentially many local optima, relying on a single OBL technique may not be sufficiently effective. In fact, if the peaks are close together, relying on a single OBL technique may not be able to prevent entrapment in local optima. Addressing this issue, assembling a sequence of OBL techniques into meta-heuristic algorithm can be useful to enhance the overall search performance. Based on a simple penalized and reward mechanism, the best performing OBL is rewarded to continue its execution in the next cycle, whilst poor performing one will miss cease its current turn. This paper presents a new adaptive approach of integrating more than one OBL techniques into Jaya Algorithm, termed OBL-JA. Unlike other adoptions of OBL which use one type of OBL, OBL-JA uses several OBLs and their selections will be based on each individual performance. Experimental results using the combinatorial testing problems as case study demonstrate that OBL-JA shows very competitive results against the existing works in term of the test suite size. The results also show that OBL-JA performs better than standard Jaya Algorithm in most of the tested cases due to its ability to adapt its behaviour based on the current performance feedback of the search process.

INDEX TERMS Opposition based Learning, Adaptive Selection, Jaya Algorithm

I. INTRODUCTION

Optimization relates to the process of finding one or more best solutions that either minimize or maximize the return on investment. Practically, finding the best solution(s) can not be guaranteed when the search spaces are very large. As a compromise, good enough solution(s) often suffice given the enormous costs involved to deal with combinatorial explosion problem.

To-date, meta-heuristic based algorithms are often sought for to deal with combinatorial explosion problem. In the field of software testing, many research adopts meta-heuristic algorithms as the basis of dealing with combinatorial explosion problem (e.g. Simulated Annealing (SA) [1], Genetic Algorithm (GA) [1, 2], Ant Colony Algorithm (ACA) [2], Particle Swarm Optimization [3], Harmony Search (HS) [4], Cuckoo Search (CS) [5, 6] and Flower Pollination Algorithm (FPA) [7]) related to t-way test suite generation. The t-way test suite generation (where t indicates the interaction strength), involves finding an optimized set of test cases that covers the t-way interaction strength. Many reported

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3055367, IEEE Access

test results indicate that t-way test suite is as good as exhaustive testing [8, 9].

Over the last 10 years, many new meta-heuristic algorithms have been developed, often, disguised by some new inspirations and mathematical formulation. Despite these so-called new inspirations and formulation, the fact remains the same[10]. The performance of any meta-heuristic algorithm is dependent on two core parts: intensification (local search) and diversification (global search). Intensification explores the promising neighbouring regions in the hope to find better solutions. On the other hand, diversification ensures that all regions of the search space have been visited, which enables the algorithm to jump out of any local optimum[11].

More specifically, the performance of meta-heuristic algorithms is highly dependent on:

a) The fine balance between the intensification and diversification. Too much intensification may result in the quick loss of diversity in the population which increases the possibility to make the algorithm being trapped in a local optimum. Aggressive diversification may lead to inefficient search and slows down the overall search performance [12, 13].
b) The operators or components that used for performing the intensification and diversification such as selection mutation, and crossover in Genetic Algorithm (GA) or local and global pollinations in Flower Pollination Algorithm (FPA)[14].

To enhance the search performance, many researchers have turned in to Opposition-based Learning (OBL) technique[15-18]. The main strength of OBL is the fact that alternative candidate solutions can be generated from one or more opposite directions, thus, ensuring sufficient coverage of the search space. Recently, many OBL techniques have been established in the literature including the Standard-OBL, General-OBL, Quasi Reflection-OBL, Centre-OBL and Optimal-OBL[15]. The OBLs have been integrated into many soft computing algorithms such as optimization methods[16], Artificial Neural Networks (ANN) [17], Reinforcement Learning (RL) [19], and Fuzzy System[18], to name a few. Meta-heuristic algorithm such as GA[20], SA[21], PSO[22], Biogeography-based Optimization (BBO)[23], HS[24], Gravitational Search Optimization (GSO) [25], Ant Colony System (ACS)[26], and Group Search Algorithm (GSA)[27], have been known to utilize the concept of OBL to enhance the performance of their search capabilities [28]. Meanwhile, in the field of Artificial Neural Network, the OBLs are used to enhance the training in Backpropagation through time (BPTT) neural network[17]. For the same purpose, the OBLs have also been adopted in Reinforcement Learning [19] to solve the problem of delayed reward in reinforcement learning.

Although proven useful, much existing integrations of OBL into meta-heuristic algorithms have been based on a single technique. If the search space contains many peaks with potentially many local optima, relying on a single OBL technique may not be sufficiently effective. If the peaks are close together, a single OBL technique may not be able to prevent entrapment in local optima. Addressing this issue, ensembling a sequence of OBL techniques into the meta-heuristic algorithm can be useful to enhance the overall search performance. Based on a simple penalized and reward mechanism, the best performing OBL is rewarded to continue its execution in the next cycle, whilst poor performing one will miss cease its current turn. This paper presents a new adaptive approach of integrating more than one OBL techniques into Jaya Algorithm, termed OBL-JA. Unlike other adoptions of OBL which use one type of OBL, OBL-JA uses several OBLs and their selections will be based on each individual performance. The Jaya Algorithm has been chosen because it is free of parameter and easy to implement.

Moreover, mixed results show that the capability of existing t-way strategies is still limited as there is no single strategy appears to be superior in all configurations considered [8, 29]. The effort to address the aforementioned shortcomings is justified through the search for a new strategy that takes the new breed of newly developed meta-heuristics algorithms into account.

Given such prospects, this paper proposes a new t-way testing strategy based on adaptive Opposition-based Learning Jaya Algorithm called OBL-JA, for t-way test suite generation. Our contributions can be summarized as follows:

- First, this paper presents a new adaptive approach of Jaya Algorithm based on Opposition-based Learning, called OBL-JA. Unlike other variants of OBL, the proposed approach uses several OBLs and the selection mechanism of OBLs will be based on current performance whereas other OBLs use only one type of OBL. By doing so, OBL-JA ables to achieve a fine balance between intensification and diversification, since OBL-JA adapts dynamic selection mechanism between different OBL operators which each has different capabilities.
- Second, this paper proposes a new t-way testing strategy based on OBL-JA for generating t-way test suite that can add a new value in the domain of software testing. The proposed strategy is compared with different t-way testing strategies. Here, two experiments have conducted; the first experiment measures the percentage use of each OBL operator in OBL-JA. while the second experiment measures the exploration and exploitation of the proposed strategy.

The rest of this paper is structured in the following manner. Section 2 gives an overview of t-way testing and its theoretical background. Then, section 3 provides reviews of existing strategies. Detailed review on OBLs and its variants are provided in section 4. Section 5 presents the design of the proposed strategy. Experiment and discussion of results are elaborated in section 6. Lastly, section 7 concludes the work along with the recommendations for future work.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3055367, IEEE Access

II. OVERVIEW ON T-WAY TESTING

A. T-way Test Suite Generation

t-way testing is a sampling technique used for generating representative test cases that can for testing software/hardware systems overall. The idea behind the t-way testing is that the tester doesn't have to test all inputs and output combinations, instead, the tester needs to meet some level of coverage such that every t combinations are covered by the test cases.

To illustrate how t-way testing can reduce the size of test cases, consider the online payment system. It allows the electronic transfer of the many in which the user have to fill out an online payment form with required information and submit to the merchant's website. In this illustrative example, there are six inputs or parameters need to be keyed and submitted to merchant's website which are selected payment method, card number, name on card, expiration, and card CVV, as shown in FIGURE 1. Five payment methods are supported by the system which are "Visa Card", "Master Card", "American Express", "Discover", and "PayPal". The fields "Name-On-Card" and "Card-Number" accept one string value for each while "Expiration-Date" are two input values MM for months and YY for years from 16 to 31. Card CVV parameter accepts one input value.

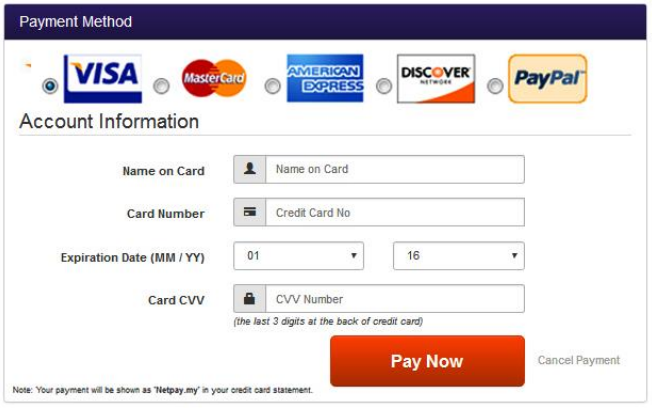

FIGURE 1. Example of Online Payment

Ideally, testing this system requires 900 test cases (5×1×1×12×15×1) which are exhaustively covered all combinations of the six parameters' values, however, testing all the combinations especially for the complex system is impractical. Turning to two-way test suite can reduce the test cases to 180 test cases, thereby saving 80% in time, effort and costs. Based on some studies, the 180 test cases generated using two-way testing (interaction coverage t = 2) can detect 93% of software failures, while 98% of failures can be detected if all three-way testing is applied. The same study shows that the rate of fault detection can reach 100% if the interaction coverage strength is between 4 and 6 [30-34].

B. Theoretical Background

Test suite (T) is an $n \times m$ array of n rows of test cases. Each test case is combinations of parameters' values. Covering array (CA) is a mathematical notation that is used to describe the t-way test suite [35, 36]. The notation CA(N, t, v^p) represents the uniform covering array where p denotes number of parameters, v denotes the values of the parameter, t donates the level of interaction strength. For example, CA(18; 2, 3^{13}) consists of 18 rows of test cases that are generated from 13 columns of parameters with three values for each parameter. If the covering array is not uniform and values of the parameters are not the same, it is represented by MCA(N, t, v_1^{p1} v_2^{p2} v_3^{p3} .....v_j^{pj}) termed as mixed CA. MCA (12, 3, 2^3 3^1) represents a covering array with 12 final test cases, generated for the system with 3 2-valued parameters and 1 3-valued parameter.

III. RELATED WORK

In the domain of software testing the existing t-way testing strategies can be characterized into two main approaches: Algebraic and Computational Approaches [4, 37]. Algebraic approach often generates the test sets without considering any combinations because generating the test set is done directly using some lightweight computations. Strategies of this approach include t-way covering Array(CA), orthogonal Latin squares (OLS), and test configuration (TConfig). However, the limitation of this approach is that the algebraic based strategies are often restricted to small configurations [38, 39]. In the other hand, generating the test suite in computational approaches is based on greedy algorithms such to cover the maximum number of interaction combinations. Tools and strategies of this approach generate the test cases either using the One Parameter at a Time (OPT) or One Test at a Time (OTT) approach.

OPT strategies generate a complete test cases with t size of parameters, then horizontally adding one parameter per iteration till all the combinations are covered. The best example of this approach is in-parameter-order (IPO) strategy and its variants [40, 41].

OTT strategies iteratively generate one complete test case per until all combination of the values is covered. An example of these approaches is the automatic efficient test generator (AETG)[42]. Based on the concept of AETG, various strategies have been developed such as GTWay [43], Jenny [44], TConfig [45], and WHITCH [46].

Due to its efficient, many researchers adopt meta-heuristic algorithms such as TS, SA, ACA, GA, HS, FPA, and CS in generating t-way test cases. In general, meta-heuristic based t-way strategies use the algorithm as core implantation for generating the test suite. Most of the meta-heuristic based t-way strategies generating the test suite using OTT. The strategy uses the meta-heuristic algorithm for generating one test per iteration then add the generated test case into the final test case. Then this procedure is repeated until all combinations are covered. In the literature review, we can recognize three categories of meta-heuristic based strategies. The first category uses a single meta-heuristic algorithm as the search engine for the test case. Example of this category

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3055367, IEEE Access

includes SA[1], GA[1, 2], ACA[2], PSO[3], HS [4], FPA[7], Whale Optimization Algorithm [47] and CS [5]. The second category uses adaptive or hybridization of meta-heuristics algorithms as the search engine. Example of this category involve high-level hyper-heuristic (HHH) [48], elite-FPA[49], Learning-CS [50], Modified ABC [51], Hybrid HS with Grey Wolf Optimizer [52], Self-adaptive FPA, Hybrid ABC [53], and Improved-JA[54].

Based on the above-mentioned review, most of the existing strategies based on single meta-heuristic algorithms. Only a few works have been done using hybridization or adaptive meta-heuristic algorithms. Another point worth to mention is that most of the existing strategies rely on some parameters and need to be tuned. In this research, we propose new t-way testing based on adaptive OBL-Jaya Algorithm which is free of parameters. The strategy adapts OBL operator to enhance its search capabilities.

IV. PROPOSED STRATEGY

The proposed strategy can be considered as two levels of optimization; the first level uses Jaya algorithm as core implementation, while the second level adopts different OBL operators, including standard-OBL, General-OBL, Quasi Reflection-OBL, Quasi Reflection-OBL, Centre-OBL and Optimal-OBL, to generate the opposition of the current population.

A. Original Opposition-based Learning and its Variants

1. Opposition-based Learning

In general, the idea of basic Opposition-based Learning (OBL) is that corresponding opposite if the current solution maybe is better than the current solution itself. It attempts to provide a better chance of finding a solution x^* from current solution x as follows:

$$x^* = a + b - x$$

where a and b are the lower and higher boundaries of x.

2. General Opposition-based Learning

General Opposition-based Learning (OBL-G) [55] uses the consent of basic OBL and Cauchy mutation (i.e. random weight), which can help trapped solution to jump out of local minima.

$$x^* = w * (a + b - x)$$

where w is a random number ∈ [0,1]

3. Quasi-Opposition Based Learning

Quasi-Opposition Based Learning (OBL-Q) [56] generates a random point between the two inverse solutions (i.e. the centre point and OBL point of x). OBL-Q is defined by:

$$x^* = \begin{cases} rand(C, a + b - x) & x < C \\ rand(a + b - x, C) & x > C \end{cases}$$

$$C = \frac{a + b}{2}$$

4. Quasi Reflection Opposition based Learning

Quasi Reflection Opposition based Learning (QR-OBL) [57] is an extension of quasi Opposition based Learning, which represents a point between the center point and x which can define by:

$$x^* = \begin{cases} rand(x, C) & x < C \\ rand(C, x) & x > C \end{cases}$$

$$C = \frac{a + b}{2}$$

5. Current Optimum Opposition based Learning

Another version of OBL is Current Optimum Opposition based Learning (OBL-O) [58] which uses the search information of the current best solution. The OBL-O is defined by:

$$x^* = 2 \times x_{best} - x$$

6. Centroid-Opposition based Learning

Centroid-Opposition based Learning (OBL-C) [59] replace Current Optimum in OBL-O by centroid opposition, which can be computed by:

$$x^* = 2 \times C - x$$

$$C = \frac{\sum_{i=1}^{n} x_i}{N}$$

where N is the population size.

B. Original Jaya Algorithm

Jaya Algorithm(JA) [60] is one of the recent meta-heuristic algorithms, designed for solving general optimization problems. The idea of JA is that potential solution should be based on the best solution and avoid the worse solution. Thus, JA needs only the best and worse solutions to generate a new solution. For generating a new solution $X'_{i,j}$, the following equation is used:

$$X'_{i\,i} = X_{i.i} + Rnd_1 (X_{best.i} - |X_{i.i}|) - Rnd_2 (X_{\text{worst,j}} - |X_{i,j}|)$$

where $X_{i,j}$ is the current solution, X_{best} is the best solution and X_{worst} is the worst solution. FIGURE 2 summarizes the Jaya algorithm.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3055367, IEEE Access

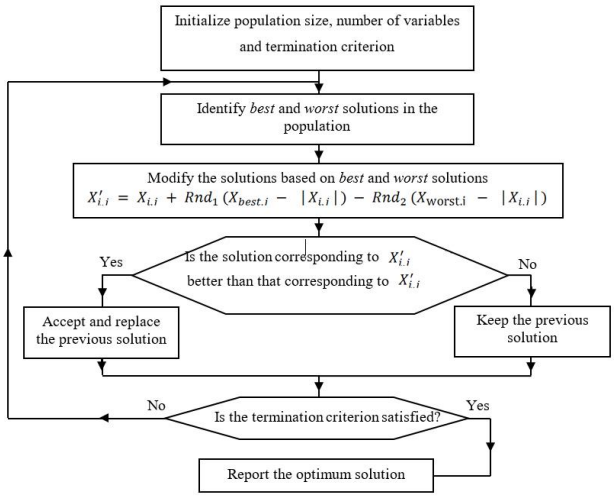

FIGURE 2. **Original Jaya Algorithm**

C. *Adaptive Jaya Algorithm based on Opposition-based Learning for Test Suite Generation*

The proposed strategy utilizes Jaya Algorithm (JA) as core implementation meta-heuristic algorithm to generate optimal t-way test suite. The OBLs are included in OBL-JA to accelerate the convergence of the search process. OBL-JA use the OBL operators for generating opposite population. The current populations and their opposite populations are evaluated simultaneously, hence, the selection of OBL to be used in the next iteration is based on obtained results. Therefore, the selection mechanism used in OBL-JA can be seen as a switch that turns OBLs on or off and switches between the OBLs list based on the performance of the algorithm overall. FIGURE 3 illustrates the flow of OBL-JA. The proposed strategy can be seen as two levels of optimizations; the first level uses Jaya algorithm, while the second level adopts different OBL operator to generate the opposition of the current population.

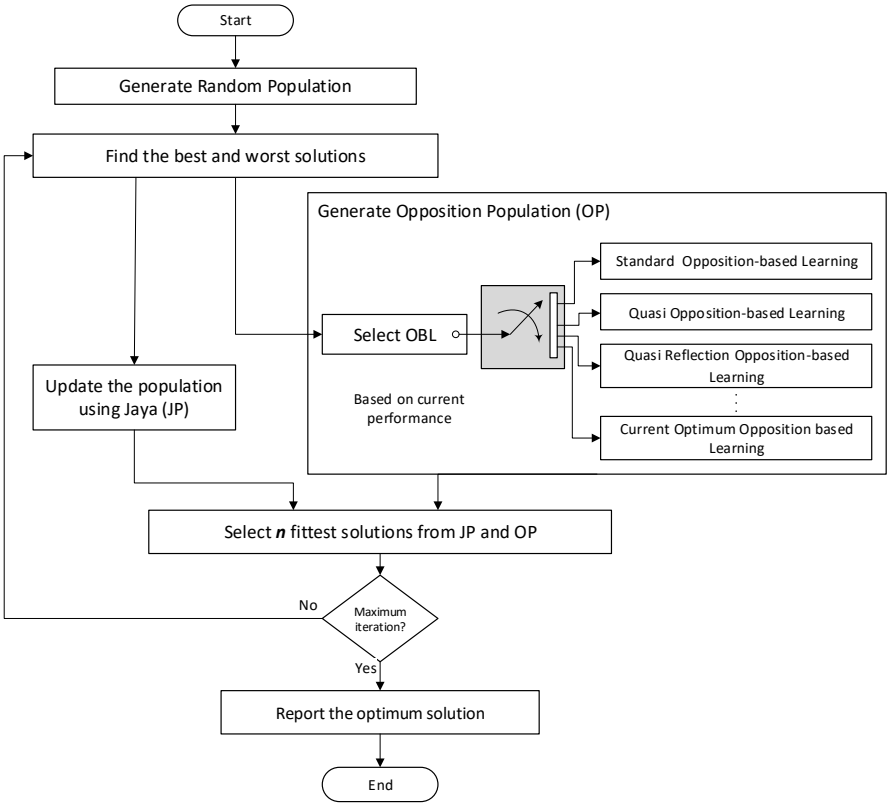

FIGURE 3. **Illustrates the Process of OBL-JA**

In order to generate t-way test suite, the strategy starts generating all t possible interaction of the inputs, which represent the search space to be added into the interaction list. For instance, if $t = 3$, the 3 combinations for 4 inputs (i.e. A, B, C, and D) with 2 values for each, are ABC, ABD, ACD, and BCD. Then for each combination, all possible 3 interactions are generated as follows (refer to FIGURE 4):

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3055367, IEEE Access

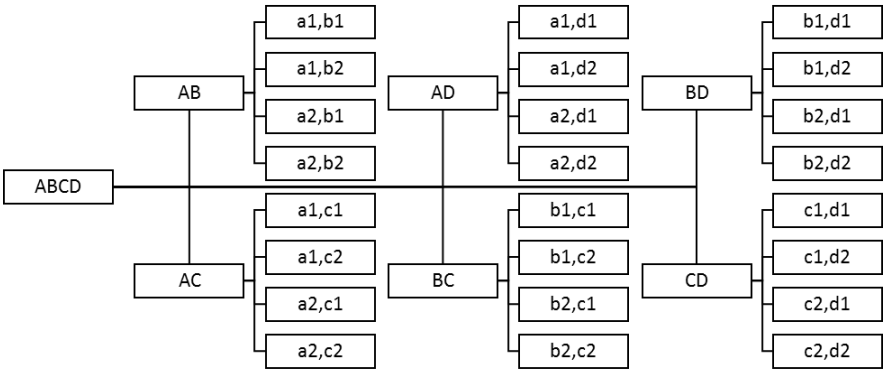

FIGURE 4. All 2-interaction possibilities for four inputs A, B, C and D

The next step of OBL-JA is to find the smallest number of test cases that cover all those interaction possibilities of the inputs. In OBL-JA, each solution represents one test case. OBL-JA starts generating a population of solutions individually, using Jaya Algorithm (JP) and its opposite ones (OP) simultaneously. Then elite solutions from JP and OP are selected to form the next population as shown in FIGURE 3. The complete step for OBL-JA includes finding the optimal test case, that is, the best solution with the highest weight. Here, the solution's weight is the number of t-combination elements x_i that can be cover by solution, which is defined by:

$$\text{Maximize } f(x) = \sum_{1}^{N} x_i$$

where, $f(x)$ is a fitness function to be optimized that captures the weight of the test case x, and x_i are covered t-combinations. The population iteratively subjects to improvement process until the termination condition is met (i.e. reach the maximum number of improvement). The best solution will be selected and added to the final test suite and the covered interaction elements are removed from the t-combinations list. The whole process is repeated until all t-combinations are covered. The complete pseudocode of OBL-JA is presented in FIGURE 5.

V. EXPERIMENTS AND DISCUSSION

In order to evaluate the efficiency of the proposed strategies, first, the variants of OBL based Jaya Algorithm are evaluated against themself. Then, the proposed OBL based Jaya Algorithm is compared with existing test generation strategies. The results are displayed using tables and graphs. The experiments were performed on Core i7-3770 CPU@ 3.40 GHz - 3.40 GHz, Windows 7 professional machine. We have adopted the tune Jaya Algorithm parameters based on existing study for generating test suite generation. The results are depicted in TABLES 2 to 10, and FIGURES 6 to 9. For statistical significance, OBL-JA is executed twenty times noting both the average and the best results. Each cell indicates the minimum test suite size obtained by existing strategies. Cells marked by star (*) denote the best test size obtained by the corresponding strategy, while cells marked as bold font cell denote the best test size obtained by OBL-JA compared with standard Jaya Algorithm. Cells marked as NA denote unavailability of the results in the literature such as the results of mAETG, AETG, HHH, and CS.

A. Parameter Adjustment

TABLE 1 depicts the parameters that are adopted for the meta-heuristic strategies [4, 6, 14, 48]. Regarding OBL-JA's parameter setting, the common two parameters, population size and iteration number, are tunned in order to select optimal values that can lead to best results. For tuning of these parameters, two well-known covering arrays, CA (N; 2, 4^6) and SCA (N; 3, 9) [61], have been used. First, in order to determine optimal test suite size, we try different values for population size and are iteration.

Concerning the population size and iteration, from the results shown in TABLE 1, TABLE 2, and FIGURE 6, it is observed that using large value of population size may lead to better results, and using too little value may lead to poor results. With increasing the number of population up to 30, the performance of OBL-JA improves, however, the high value of population size (i.e., equal to 500) does not necessarily give better size of the test suite as shown in FIGURE 6. Hence, for population size, the best results are obtained when the population is between 30 and 100. As well, we can observe that as the iteration value increases, the best result obtained is also getting better. The best result obtained is when the iteration value varies from 300 to 500.

B. Analyzing the behaviour of OBL-JA

In order to analyze the behaviour of OBL-JA and evaluate the effect of introducing the OBLs into standard, two experiments have been conducted. In fact, the performance of OBL-JA is heavily based on selected OBL which adopts a selection mechanism to switch between OBL operators. Hence the first experiment depicts the variation of the obtained results from each implemented OBL operator. This experiment is to evaluate the effect of introducing the different OBL operator on the obtained results. Considering CA-1: CA(N, 2, 10^2) and CA-2: MCA(N, 2, 10^2), FIGURE 7 illustrates the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3055367, IEEE Access

percentage distribution of each OBL. The figure shows the utilization of OBL operators by OBL-AJ. The figure shows that OBL-JA prefers the OBL-QR search operator with 23% and 21% for CA-1 and CA-2, respectively. However, the other operators show competitive results compared with standard Jaya algorithm. TABLE 4 shows the comparison of different variants of OBL based Jaya algorithm with itself. The table shows the results of implementing each OBL independently. The results of both OBL-QR and OBL-O show its superior to outperformance other variants of OBL operators in the two problems, while the worst results go to OBL-C.

```
Input: (P, v, t):  Set of parameters, its values and
                   interaction strength level.
       OBL:        list of OBLs
Output:            Final_TC final test cases list
Generating Interaction Tuples of (P, v, t)
Let Final_TC be a set of final test cases;
Generate n initial population randomly
  while Interaction tuples are not cover do
    while t < MaxIteration do
      // Perform  Jaya Algorithm
      for i = 1 : n (all population)
          Generate test case Tc using Jaya equation
          Generate test case Tc-OBL using OBL[i]
      end for
      // Evaluate the new population
        Select n elite solutions from Tc and Tc-OBL
        Evaluate new solutions
        if new solutions are better, then
            Continue with same OBL[i]; i=i;
        else
            Move to next OBL[i]; i =  i + 1
        end if
      Find the current best gbest
    end while
    Add gbest into Final_TC list.
    Remove covered interactions tuples.
  end while
```

FIGURE 5. OBL-JA Pseudocode

TABLE 1: Parameters for Meta-Heuristic Strategies of Interests

Algorithm	Parameter	Values
GA	Maximum iteration	1000
	Population size	25
	Best cloned	1
	Random crossover	0.75
	Tournament selection	0.8
	Mutation rate	0.03
	Max stale period	3
	Escape mutation	0.25
ACA	Maximum iteration	1000
	Number of ants	20
	Pheromone control	1.6
	Pheromone persistence	0.5
	Heuristic control	0.2
	Pheromone amount	0.01
	Initial pheromone	0.4
	Max stale period	5
	Elite ants	2
SA	Maximum iteration	1000
	Cooling schedule	0.9998
	Starting temperature	20
PSO	Maximum iteration	100
	Population size	80
	Inertia weight	0.3
	Acceleration coefficients	1.375
CS	Maximum iteration	100
	Population size	100
	Probability *ep*	0.25
FPA	Maximum iteration	300
	Population size	100
	Probability *pa*	0.25
HS	Maximum iteration	1000
	Harmony memory size	100
	Harmony memory consideration rate	0.7
	Pitch adjustment rate	0.2
HHH	Maximum iteration	100
	Population size	40
	$Tabu_{max}$	4
	Inertia weight	0.3
	Acceleration coefficients (c_1,c_2)	1.375
	Probability *p*	0.25

TABLE 2: Averages Test Suite for CA(N; 2, 4^6)

Population Size	Iteration										
	5	**10**	**20**	**30**	**40**	**50**	**100**	**200**	**300**	**500**	**700**
10	35.65	32.15	32.95	30.30	29.95	29.15	28.55	27.65	26.40	25.65	25.25
20	32.20	29.65	28.80	28.45	27.55	27.40	26.95	25.10	24.95	24.80	24.50
30	30.95	28.65	28.00	26.80	26.75	25.50	25.35	24.65	24.50	24.20	24.00
50	29.05	26.65	27.20	25.95	25.95	25.15	25.55	24.10	24.00	24.00	24.00
100	27.15	26.10	25.65	25.25	24.65	24.70	24.55	23.85	24.10	23.90	23.50
200	26.25	25.15	24.80	24.65	24.50	24.35	24.00	23.40	23.50	23.90	23.75
300	25.90	24.90	24.55	24.05	24.45	24.15	23.95	23.80	23.70	23.45	24.00
500	24.80	24.50	24.15	24.05	23.70	23.80	24.10	23.35	23.55	23.80	23.65

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3055367, IEEE Access

TABLE 3: Averages Test Suite for CA(N; 2, 3^{13})

Population Size	Iteration										
	5	10	20	30	40	50	100	200	300	500	700
10	29.2	24.8	23	22.6	22	21.8	21.4	20.4	20.4	21	20.4
20	25.4	24.4	22	21	21.2	21.2	20.4	20.6	20	20.6	20.4
30	26	23.4	21.8	21.6	21.4	21	20.6	20.2	20.4	20.4	19.8
50	23.8	22.4	21	21	21	20.4	20.2	20.6	20.2	20.2	20.8
100	23.6	21.2	21.2	20.4	20.4	20.2	20.2	20	20.4	19.8	20
200	22.4	21.4	21	21	20.4	20.4	20.2	20.6	20	19.6	20
300	21.6	20.8	20.6	20.4	20.4	20.2	20.2	20.2	20	19.8	20
500	21.8	21.2	20.4	20	19.6	20.4	20.4	20.6	20.2	19.8	20

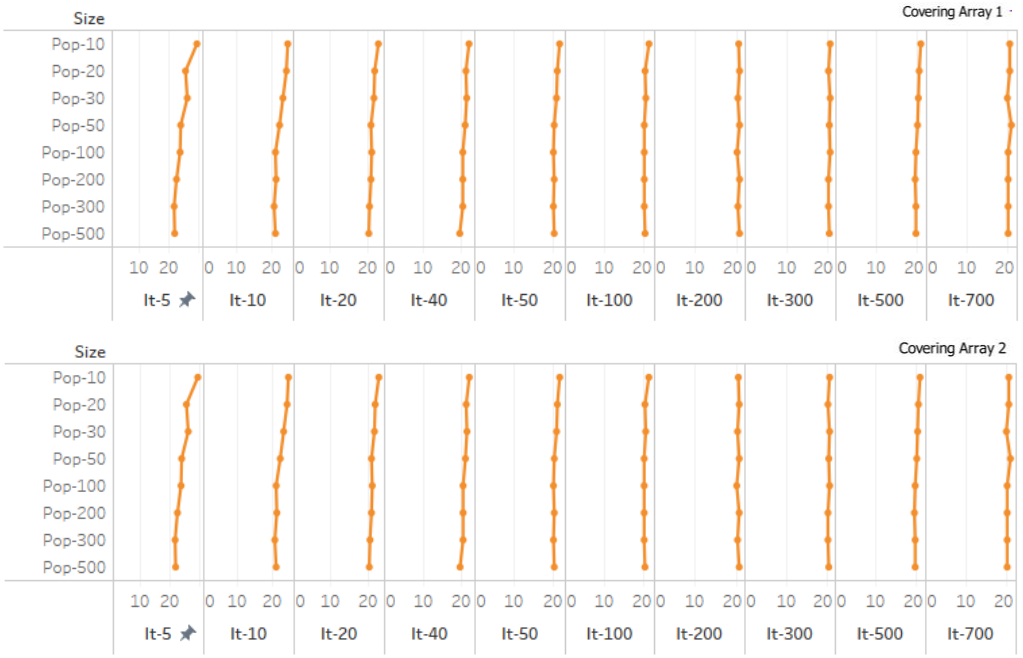

FIGURE 6. Visual representation of averages test suite for CA(N; 2, 4^6) and CA(N; 2, 3^{13})

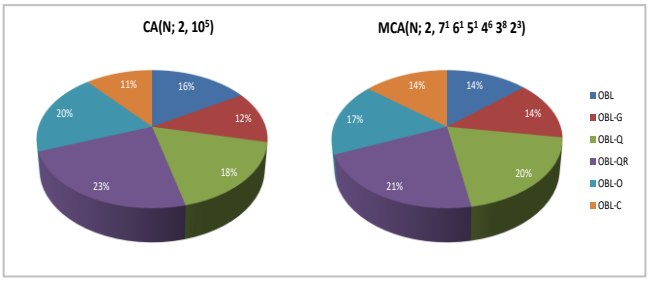

FIGURE 7. OBL Operator Normalized Percentage Distribution for CA(N; 2, 10^5) and MCA(N; 2, 7^1 6^1 5^1 4^6 3^8 2^3)

TABLE 4. Comparison of Different Variants of OBL based Jaya algorithm

Variants of Jaya	CA1: CA(N; 2, 10^5)		CA2: MCA(N; 2, 7^1 6^1 5^1 4^6 3^8 2^3)	
	Average	Best	Average	Best
OBL-S	123.6	123	54.33	53
OBL-G	122.3	121	53.83	53
OBL-Q	124.75	123	54.66	53
OBL-QR	122.8	119*	53.16	52*
OBL-O	225.5	224	53.83	52*
OBL-C	224.83	224	55.33	54
Standard Jaya	123.66	122	53.33	53
Proposed Strategy	121.3	**118***	53.16	**49***

Similar to other met-heuristic algorithms, the performance of OBL-JA is dependent on the fine balance between exploration and exploitation. Too much exploitation may result in the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3055367, IEEE Access

quick loss of diversity in the population which increases the possibility to make the algorithm being trapped in a local optimum. Aggressive exploration may lead to inefficient search and slows down the overall search performance [13]. To determine the exploration and exploitation of the proposed strategy, an experiment is conducted by measuring the hamming distance between the population of test cases with themselves which is also known as the population's diversity rate. If the distance is large, then it is exploring, otherwise, it is exploiting the search space. The following equation is used to measuring the hamming distance between two test cases (i.e. x_t^i and x_t^{i+1}):

$$\text{Distance} = \sum_{i=0}^{d} x_t^i - x_t^{i+1}$$

FIGURE 8 shows the average distance of the population at each iteration. Besides the standard Jaya and proposed strategy, the figure shows the diversity rates of other variants of OBL based Jaya algorithm based on the results of the first covering array in TABLE 4. The figure also shows that OBL-G and OBL-G obtained the lowest diversity rates which meant they tend to exploitation rather than exploration. In contrast, both of OBL-QR and OBL-JA obtained the highest rating. Comparing the standard Jaya against the OBL-JA, OBL-JA allows more diverse solutions than standard Jaya. Although obtained the highest diversity rate among other variants of OBL based Jaya Algorithm, OBL-JA still achieves a balance exploration and exploitation since it is less than the maximum diversity rate.

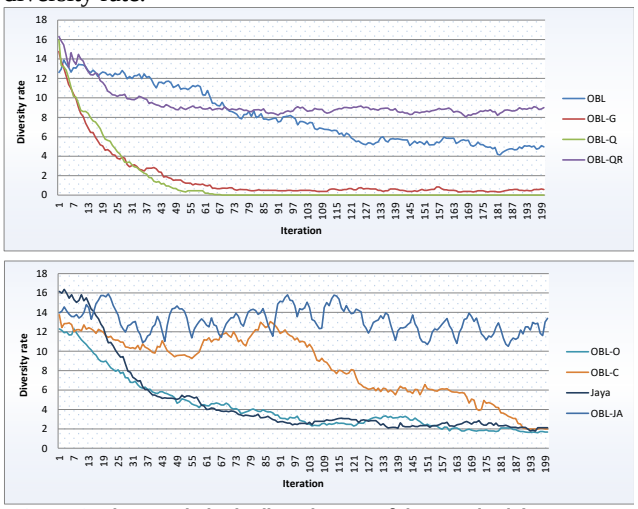

FIGURE 8: The population's diversity rate of the standard Jaya algorithm, proposed strategy, and other variants of OBL based Jaya algorithm.

C. Benchmarking with existing strategies on test sizes

To evaluate its obtained solution in terms of minimization of test suite size, OBL-JA is compared with existing meta-heuristic based t-way strategies. Our experiments are divided into four sets of comparisons as follows:

1. Comparison of OBL-JA with results of strategies published in [4, 6, 62] for different configurations system (TABLE 5): S1: CA(N; 2, 3^4), S2: CA(N; 2, 3^{13}), S3: CA(N; 2, 10^{10}), S4: CA(N; 2, 15^{10}), S5: CA(N; 2, 5^{10}), S6: CA(N; 3, 3^6), S7: CA(N; 3, 4^6), S8: CA(N; 3, 5^6), S9: CA(N; 3, 6^6), S10: CA(N; 3, 5^7), S11: MCA(N; 2, 5^1 3^8 2^2), S12: MCA(N; 2, 7^1 6^1 5^1 4^6 3^8 2^3), S13: MCA(N; 3, 5^2 4^2 3^2), S14: MCA(N; 3, 10^1 6^2 4^3 3^1).
2. Comparison of OBL-JA with existing strategies for CA (N; t, 2^{10}), *t* varied from 2 to 10 as shown in TABLE 6.
3. Comparison of OBL-JA with existing strategies for CA(N; 4, 5^p), *p* varied from 5 to 10, as shown in TABLE 7.
4. Comparison of OBL-JA with existing strategies for CA(N; 4, v^{10}),*v* varied from 2 to 7 as shown in TABLE 8 [8, 14, 63].

TABLE 5 highlights the comparative results of 14 system configurations. Overall, the table shows that SA and GA outperform other existing strategies in 7 and 6 out of 14 cell entries respectively, followed by mAETG strategy in 6 out of 14 cell entries. HHH and OBL-JA provide competitive performance with 5 cell entries for each, followed by ACA by 4 entries. PSO, HS, and CS perform the poorest with only 1 cell entry for PSO and HS, and no single entry for CS. OBL-JA contributes in terms of generating the best overall test suite size in three cases CA(N; 2, 10^{10}), CA(N; 2, 15^{10}), and MCA(N; 3, 10^1 6^2 4^3 3^1). Comparing OBL-JA and its counterparts standard Jaya Algorithm, the OBL-JA outperform Jaya Algorithm in almost all test cases except one case when CA(N; 3, 6^6). TABLE 6 highlights the case of CA (N; t, 2^{10}) where t is varied from 2 to 10. The results in the table show that most of the existing strategies are unable to produce results beyond t > 6 due to their heavy computation such as GA, ACA GA and PSO. In general, meta-heuristic-based strategies have gained the top performance among computational-based strategies. Specifically, OBL-JA has gained the first rank by obtaining 4 out of 9 cell entries and HHH and FPA gained the second rank by obtaining 2 out of 9 cell entries. CS and HS also achieved good performance with single best results out of the 9 entries. Meanwhile, ITCH, IPOG, Jenny, PICT, TConfig, TVG, GTWay and PSO do not contribute to any of the best cell entries. Comparing OBL-JA and Jaya Algorithm, OBL-JA continues its superiority and outperform Jaya Algorithm in all cases except two cases when t=2 and t=10, whereby both strategies produce the same test size.

TABLE 7 reports the results for CA(N; 4, 5^p) where P is varied from 5 to 12. GTWay outperforms other strategies in 4 out of 8 cell entries while OBL-JA comes second and outperforms other strategies in 2 entries, followed by HHH and FPA with 1 entry for each. In fact, GTWay uses backtracking concept which almost generates all possible solutions using recursion. Thus the time consuming of GTWay is usually exponential or worse. Concerning the performance of OBL-JA and Jaya Algorithm, OBL-JA is still superior to the standard Jaya Algorithm, however, Jaya Algorithm outperforms OBL-JA in two cases when *p*=7 and *p*=9.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3055367, IEEE Access

TABLE 5: Comparison with Existing Strategies for Different CA and MCA Configurations

No.	mAETG	AETG	ACA	SA	GA	HS	PSO	CS	HHH	FPA	Jaya		OBL-JA	
											Avg	best	Avg	best
S1	**9***	**9***	**9***	**9***	**9***	**9***	**9***	**9***	**9***	**9***	9.97	9*	9.93	9*
S2	17	**15***	17	16	17	18	17	20	17	18	21.16	20	18.09	18
S3	NA	NA	159	NA	157	155	NA	NA	NA	153	160.44	159	150.1	**150***
S4	NA	NA	NA	NA	NA	342	NA	NA	NA	NA	337.6	289	277.8	**276***
S5	NA	NA	NA	NA	NA	43	45	NA	**42***	**42***	47.7	45	46.8	**42***
S6	38	47	**33***	**33***	**33***	39	42	43	**33***	44	47.2	43	43.6	38
S7	77	105	**64***	**64***	**64***	70	102	105	**64***	101	110.6	105	105.56	102
S8	194	NA	**125***	152	**125***	199	NA	NA	NA	195	209.54	205	200.45	197
S9	330	343	330	**300***	331	336	338	350	325	NA	341.6	332	338.17	336
S10	218	229	218	**201***	218	236	229	233	217	220	232.2	231	222.37	220
S11	20	19	16	**15***	**15***	20	NA	21	20	21	23.8	22	21.91	20
S12	44	45	**42***	**42***	**42***	50	48	51	48	NA	53.68	53	50.33	48
S13	114	NA	106	100*	108	120	NA	NA	**100***	118	126.67	123	123.8	119
S14	377	NA	361	360	360	378	385	393	382	NA	391.11	367	369.30	**355***

TABLE 6. Comparison with Existing Strategies Using CA(N; t, 2^{10}), t varied from 2 to 10

t	Computational-based Strategies							Meta-heuristic-based Strategies								
	IPOG	ITCH	Jenny	PICT`	TConfig	GTWay	TVG	PSO	HS	HHH	CS	FPA	Jaya		OBL-JA	
													Avg	best	Avg	best
2	10	**6***	10	NA	9	NA	10	8	7	8	8	8	8.77	**8**	8.62	8
3	19	18	18	NA	20	NA	17	17	16	16	16	16	18.43	17	17.9	**15***
4	49	58	39	NA	45	NA	41	37	37	**36***	**36***	35	40.66	38	40.47	37
5	128	NA	87	NA	95	NA	84	82	81	79	79	81	87.44	88	83.9	**74***
6	352	NA	169	NA	183	NA	168	158	158	**153***	157	158	165.93	161	165.64	159
7	NA	NA	311	NA	NA	NA	302	NA	298	NA	NA	**292***	302.43	297	307.6	**292***
8	NA	NA	521	NA	NA	NA	514	NA	498	NA	NA	500	518.6	508	517.5	497
9	NA	NA	788	NA	NA	NA	651	NA	**512***	NA	NA	592	709.0	639	692.0	575
10	NA	NA	1024	NA	NA	NA	NA	NA	**1024***	NA	NA	**1024***	1024.0	**1024***	1024.0	**1024***

TABLE 7. Comparison with Existing Strategies CA(N; 4, 5^P), P varied from 5 to 10.

P	Computational-based Strategies								Meta-heuristic-based Strategies								
	IPOG	ITCH	Jenny	PICT	TConfig	TVG	GTWay	CTE-XL	MIPOG	PSO	HSS	HHH	CS	FPA	Jaya		OBL-JA	
															Avg	best	Avg	best
5	908	837	810	773	849	731	**625***	NA	779	779	751	746	776	784	786.3	779	787.3	776
6	1239	1074	1072	1092	1128	1027	**625***	NA	1001	1001	990	967	991	988	1006.71	1004	994.6	987
7	1349	1248	1279	1320	1384	1216	1125	NA	1209	1209	1186	**1151***	1200	1164	1178.6	1175	1188.6	1183
8	1792	1424	1468	1532	1595	1443	1384	NA	1417	1417	1358	1320	1415	1329	1372.2	1364	1336.86	**1322***
9	1793	1578	1643	1724	1795	1579	1543	NA	1570	1570	1530	1483	1562	1478	1528.8	1525	1481.4	**1474***
10	1965	1791	1812	1878	1971	1714	1643	NA	1716	1716	1624	1635	1731	**1605***	1729.53	1693	1644.23	**1616**
11	2091	1839	1957	2038	2122	1852	**1722***	NA	1902	1902	1860	1784	2062	1739	1796.67	1745	1788.51	1761
12	2285	1964	2103	NA	2268	2022	**1837***	NA	2015	2015	2022	1915	2223	1879	1981.52	1978	1924.67	1863

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3055367, IEEE Access

TABLE 8: Comparison with Existing Strategies CA(N; 4, v^{10}) with v varied from 2 to 7.

V	Computational-based Strategies									Meta-heuristic-based Strategies								
	IPOG	ITCH	Jenny	PICT	TConfig	TVG	GTWay	CTE-XL	MIPOG	PSO	HSS	HHH	CS	FPA	Jaya		OBL-JA	
															Avg	best	Avg	best
2	49	58	39	43	45	40	46	NA	43	34	37	36	**28***	36	43.16	41	41.46	35
3	241	336	221	231	235	228	224	NA	217	213	211	**207***	211	211	217.5	214	213.4	209
4	707	704	703	742	718	782	**621***	NA	637	685	691	668	698	661	707.7	704	672.62	670
5	1965	1750	1719	1812	1878	1917	1714	NA	1643	1716	1624	1635	1731	1605	1701.45	1644	1592.0	**1592***
6	3935	NA	3519	3735	NA	4159	3514	NA	3657	3880	3475	3405	3894	3354	3387.15	3355	3345.51	**3321***
7	7061	NA	6462	NA	NA	7854	6459	NA	**5927***	NA	6398	6412	NA	6205	6335.56	6198	6281.35	6177

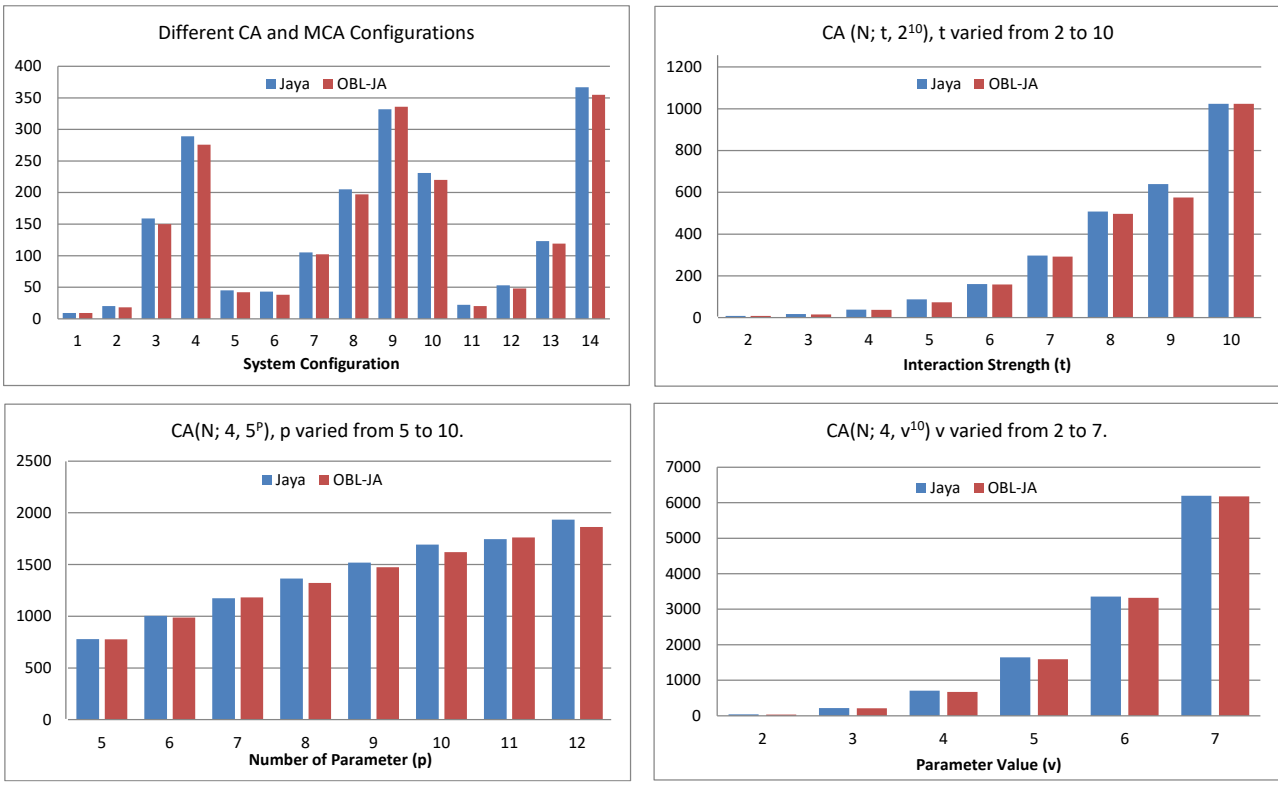

FIGURE 9. illustration of the Comparison between OBL-JA and standard Jaya Algorithm.

As for comparative experiment involving CA(N; 4, v^{10}) with v is varied from 2 to 7 in TABLE 8, OBL-JA outperforms the existing strategies in 2 out of 6 cell entries, while GTWay, MIPOG, CS and HHH come as the runner up with only 1 best entry. IPOG, ITCH, Jenny, PICT, TConfig, TVG, CTE-XL, PSO, and HSS perform the poorest with no single best cell entry. In this experiment, the standard Jaya algorithm fails to outperform the proposed strategy in any case.

FIGURE 9 illustrates the comparison of OBL-JA against Jaya Algorithm for TABLE 6 till TABLE 8. The comparison shows that OBL-JA performs better than standard Jaya Algorithm in most of the cases. OBL-JA is able to generate better results due to its ability to adapt its behaviour based on the problem itself. As state earlier, the performance of any meta-heuristic based strategies depends heavily on their exploration and exploitation. OBL-JA utilizes the search capabilities of OBL operators since each OBL has its own searching capability, therefore it is able to switch from one OBL to another based on addressing the problem.

D. Statistical Analysis

In order to analyze and verify our findings, this section presents a statistical analysis. Multiple comparisons for all obtained results are conducted. Wilcoxon signed-rank tests with Bonferroni–Holm correction are used to find whether the proposed strategy presents statistical difference with regards to the existing strategies.

Post-hoc Wilcoxon Signed Rank Test technique is used to analyze the significance of each pair of strategies. The Wilcoxon test is a non-parametric analysis technique can be

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3055367, IEEE Access

used to compare two sets of ordinal data that are subjected to different conditions. Wilcoxon test statistic is calculated and converted into a conditional probability *P-value*. A small *P-value* means that it is strong evidence to reject the null hypothesis H_0 (i.e. there is no difference between two strategies' results) in favor of the alternative hypothesis.

In this test, the OBL-JA is compared with each existing strategy, separately; to test if there is a significant difference between the produced results of the proposed strategy and other strategies. Here, we have two different hypotheses null hypothesis (H_0) and the alternative hypothesis (H_1). H_0 indicates that there is no difference between the two strategies' results, while H_1 indicates that there is a difference between the two strategies' results. In other words, H_1 indicates that obtained test size using OBL-JA is less than each individual strategy.

Since we are dealing with multiple comparisons, it is more likely to face Type I errors which is the rejection of a true null hypothesis[64]. To control such effect, there is a need for adjusting the rejection criteria for each individual test. Here, Bonferroni–Holm correction is adopted for each comparison level. By using Bonferroni–Holm correction, sorted p-values are compared with adjusted alpha[65]. TABLE 9 depicts the Post-hoc Wilcoxon signed-rank test with Bonferroni–Holm correction for the six OBL's variants along with standard Jaya Algorithm and OBL-JA. The statistical results in the table are for nine systems. The results show that in all pair comparisons, the values of Asymp. Sig. (2-tailed) of Post-hoc Wilcoxon Rank-Sum Tests are less than Bonferroni–Holm correction, which means statistically there is a significant difference in the results of each pair with superior of the proposed strategy compared to all pairs except for OBL-QR variant. OBL-QR variant shows competitive results compared with the proposed strategy, and this reinforces the obtained results in TABLE 4. TABLE 10 shows statistical analyses for the results in TABLE 5 until TABLE 8. Strategies with one or more NA entries such as mAETG, AETG, SA, ACA, GA are ignored. The statistical results show that in most comparisons the proposed strategy the null hypothesis is rejected with a significant difference. Although no statistical difference is shown in some comparisons such as OBL-JA vs ITCH, OBL-JA vs HSS and OBL-JA vs HHH, the positive ranks of OBL-JA are higher than its negative ranks.

TABLE 9: Post-hoc Wilcoxon Rank-Sum Tests with Bonferroni–Holm correction for OBL's Variants

Pairs	Ranks				Asymp. Sig. (2-tailed)	Bonferroni–Holm correction	Conclusion
	Negative Ranks	Positive Ranks	Ties	Total			
OBL-JA vs OBL-S	9	0	0	9	0.042	0.00714	Reject the null hypothesis $\mathbf{H_0}$
OBL-JA vs Jaya	9	0	0	9	0.042	0.00834	Reject the null hypothesis $\mathbf{H_0}$
OBL-JA vs OBL-G	7	0	2	9	0.043	0.01	Reject the null hypothesis $\mathbf{H_0}$
OBL-JA vs OBL-Q	4	3	2	9	0.043	0.0125	Reject the null hypothesis $\mathbf{H_0}$
OBL-JA vs OBL-O	9	0	0	9	0.043	0.0167	Reject the null hypothesis $\mathbf{H_0}$
OBL-JA vs OBL-C	9	0	0	9	0.043	0.025	Reject the null hypothesis $\mathbf{H_0}$
OBL-JA vs OBL-QR	9	0	0	9	0.068	0.05	Retain the null hypothesis $\mathbf{H_0}$

TABLE 10: Post-hoc Wilcoxon Rank-Sum Tests with Bonferroni–Holm correction for Table 5 till 8

Pairs	Ranks				Asymp. Sig. (2-tailed)	Bonferroni–Holm correction	Conclusion
	Negative Ranks	Positive Ranks	Ties	Total			
OBL-JA vs IPOG	16	0	0	16	0.0003924	0.0055	Reject the null hypothesis $\mathbf{H_0}$
OBL-JA vs Jenny	16	0	0	16	0.0003924	0.00625	Reject the null hypothesis $\mathbf{H_0}$
OBL-JA vs TVG	16	0	0	16	0.0003937	0.00714	Reject the null hypothesis $\mathbf{H_0}$
OBL-JA vs TConfig	16	0	0	16	0.0006957	0.00834	Reject the null hypothesis $\mathbf{H_0}$
OBL-JA vs PSO	14	1	1	16	0.0017244	0.01	Reject the null hypothesis $\mathbf{H_0}$
OBL-JA vs FPA	11	2	3	16	0.0048742	0.0125	Reject the null hypothesis $\mathbf{H_0}$
OBL-JA vs Jaya	13	2	1	16	0.0055665	0.0167	Reject the null hypothesis $\mathbf{H_0}$
OBL-JA vs CS	13	3	0	16	0.0134856	0.025	Reject the null hypothesis $\mathbf{H_0}$
OBL-JA vs ITCH	13	3	0	16	0.0551188	0.05	Retain the null hypothesis $\mathbf{H_0}$
OBL-JA vs HSS	9	3	4	16	0.0551073	0.0125	Retain the null hypothesis $\mathbf{H_0}$
OBL-JA vs HHH	8	2	6	16	0.0254045	0.0166	Retain the null hypothesis $\mathbf{H_0}$

Note: strategies with one or more NA entries such as mAETG, AETG, SA, ACA, GA are ignored due to uncompleted data.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3055367, IEEE Access

VI. CONCLUSION

In this paper, we have proposed a new adaptive strategy for t-way test suite generation based on Jaya Algorithm(JA) and opposition-based learning(OBL) concept, called adaptive Jaya Algorithm based on Opposition-based Learning for generating the test suite (OBL-JA). The OBL-JA has been obtained by employing the concept of component grafting of different types of OBL operators, such as standard-OBL, General-OBL, Quasi Reflection-OBL, Quasi Reflection-OBL, Centre-OBL and Optimal-OBL, into the standard JA strategy. OBL-JA adapts a kind selection mechanism between OBLs based on the performance of OBL-JA. Experiment results and statistical analysis show that the OBL-JA based strategy outperforms the existing t-way strategies in many cases. The OBL-JA is also compared with standard JA in the context of t-way test suite generation. In most of the cases, the OBL-JA performs better than standard JA due to its ability to adapt its behaviour based on the problem itself. Owing to encouraging results, we are looking to use OBL-JA for global optimization problem and explore the possibilities of constraints-based software product lines.

ACKNOWLEDGEMENT

This research is funded by Ministry of Higher education - Malaysia (MOHE), under Fundamental Research Grant Scheme (FRGS) grants: “Formulation of Bi-objective Elitist Dragonfly Algorithm (BiDA) for constructing prioritized t-way test cases”, FRGS/1/2019/ICT02/UMP/02/13, and Universiti Malaysia Pahang (UMP). We thank MOHE and UMP for the contribution and supports

REFERENCES

[1] J. Stardom, *Metaheuristics and the search for covering and packing arrays*, Canada: Simon Fraser University, 2001.
[2] T. Shiba, T. Tsuchiya, and T. Kikuno, "Using artificial life techniques to generate test cases for combinatorial testing." pp. 72-77.
[3] B. S. Ahmed, K. Z. Zamli, and C. P. Lim, “Application of particle swarm optimization to uniform and variable strength covering array construction,” *Applied Soft Computing*, vol. 12, no. 4, pp. 1330-1347, 2012.
[4] A. R. A. Alsewari, and K. Z. Zamli, “Design and implementation of a harmony-search-based variable-strength t-way testing strategy with constraints support,” *Information and Software Technology*, vol. 54, no. 6, pp. 553-568, Jun, 2012.
[5] B. S. Ahmed, T. S. Abdulsamad, and M. Y. Potrus, “Achievement of minimized combinatorial test suite for configuration-aware software functional testing using the cuckoo search algorithm,” *Information and Software Technology*, vol. 66, no. C, pp. 13-29, Oct, 2015.
[6] B. S. Ahmed, T. S. Abdulsamad, and M. Y. Potrus, “Achievement of minimized combinatorial test suite for configuration-aware software functional testing using the cuckoo search algorithm,” *Information and Software Technology*, vol. 66, pp. 13-29, Oct, 2015.
[7] A. B. Nasser, A. A. Alsewari, N. M. Tairan, and K. Z. Zamli, “Pairwise Test Data Generation Based On Flower Pollination Algorithm,” *Malaysian Journal of Computer Science*, vol. 30, no. 3, pp. 242-257, 2017.
[8] K. Z. Zamli, F. Din, G. Kendall, and B. S. Ahmed, “An experimental study of hyper-heuristic selection and acceptance mechanism for combinatorial t-way test suite generation,” *Information Sciences*, vol. 399, pp. 121-153, 2017.
[9] L. Yu, F. Duan, Y. Lei, R. N. Kacker, and D. R. Kuhn, "Combinatorial Test Generation For Software Product Lines Using Minimum Invalid Tuples." pp. 65-72.
[10] X. S. Yang, *Nature-Inspired Optimization Algorithms*: Elsevier, 2014.
[11] N. Pholdee, and S. Bureerat, “Comparative performance of meta-heuristic algorithms for mass minimisation of trusses with dynamic constraints,” *Advances in Engineering Software*, vol. 75, pp. 1-13, 2014.
[12] X.-S. Yang, S. Deb, and S. Fong, “Metaheuristic algorithms: optimal balance of intensification and diversification,” *Applied Mathematics & Information Sciences*, vol. 8, no. 3, pp. 977-983, 2013.
[13] X.-S. Yang, S. Deb, and S. Fong, “Metaheuristic algorithms: optimal balance of intensification and diversification,” *Applied Mathematics & Information Sciences*, vol. 8, no. 3, pp. 977, 2014.
[14] A. B. Nasser, K. Z. Zamli, A. A. Alsewari, and B. S. Ahmed, “Hybrid flower pollination algorithm strategies for t-way test suite generation,” *PloS one*, vol. 13, no. 5, pp. e0195187, 2018.
[15] S. Gupta, and K. Deep, “A hybrid self-adaptive sine cosine algorithm with opposition based learning,” *Expert Systems with Applications*, vol. 119, pp. 210-230, 2019.
[16] M. Ventresca, and H. R. Tizhoosh, "Simulated annealing with opposite neighbors." pp. 186-192.
[17] M. Ventresca, and H. R. Tizhoosh, "Opposite transfer functions and backpropagation through time." pp. 570-577.
[18] H. R. Tizhoosh, and F. Sahba, "Quasi-global oppositional fuzzy thresholding." pp. 1346-1351.
[19] M. Shokri, H. R. Tizhoosh, and M. Kamel, "Opposition-based Q (λ) algorithm." pp. 254-261.
[20] O. Chebbi, and J. Chaouachi, "Effective parameter tuning for genetic algorithm to solve a real world transportation problem." pp. 370-375.
[21] E. Aarts, and J. Korst, “Simulated annealing and Boltzmann machines: a stochastic approach to combinatorial optimization and neural computing,” 1988.
[22] H. Huang, A. Hoorfar, and S. Lakhani, "A comparative study of evolutionary programming, genetic algorithms and particle swarm optimization in antenna design." pp. 1609-1612.
[23] D. Simon, “Biogeography-based optimization,” *IEEE transactions on evolutionary computation*, vol. 12, no. 6, pp. 702-713, 2008.
[24] X.-S. Yang, and Z. W. Geem, *Music-inspired Harmony Search Algorithm: Theory and Applications*: Springer, 2009.
[25] E. Rashedi, H. Nezamabadi-Pour, and S. Saryazdi, “GSA: a gravitational search algorithm,” *Information sciences*, vol. 179, no. 13, pp. 2232-2248, 2009.
[26] M. Dorigo, M. Birattari, C. Blum, M. Clerc, T. Stützle, and A. Winfield, "Ant colony optimization and swarm intelligence."
[27] K. M. Gates, and P. C. Molenaar, “Group search algorithm recovers effective connectivity maps for individuals in homogeneous and heterogeneous samples,” *NeuroImage*, vol. 63, no. 1, pp. 310-319, 2012.
[28] S. Mahdavi, S. Rahnamayan, and K. Deb, “Opposition based learning: A literature review,” *Swarm and evolutionary computation*, vol. 39, pp. 1-23, 2018.
[29] A. R. A. Alsewari, and K. Z. Zamli, "An orchestrated survey on t-way test case generation strategies based on optimization algorithms." pp. 255-263.
[30] A. W. Williams, and R. L. Probert, *Software components interaction testing: coverage measurement and generation of configurations*: University of Ottawa, 2002.
[31] D. R. Kuhn, R. N. Kacker, and Y. Lei, “Practical combinatorial testing,” *National Institute of Standards and Technology (NIST) Special Publication*, vol. 800, pp. 142, 2010.
[32] M. Grindal, J. Offutt, and S. F. Andler, “Combination testing strategies: a survey,” *Software Testing Verification & Reliability*, vol. 15, no. 3, pp. 167-199, Sep, 2005.
[33] K. Z. Bell, “Optimizing effectiveness and efficiency of software testing: a hybrid approach,” Doctoral dissertation, North Carolina State University, ACM, 2006.
[34] K. Z. Bell, and M. A. Vouk, "On effectiveness of pairwise methodology for testing network-centric software." pp. 221-235.
[35] K. Burr, and W. Young, "Combinatorial test techniques: table-based automation, test generation and code coverage."
[36] C. Yilmaz, M. B. Cohen, and A. A. Porter, “Covering arrays for efficient fault characterization in complex configuration spaces,” *IEEE Transactions on Software Engineering*, vol. 32, no. 1, pp. 20-34, 2006.

[37] Y. Lei, R. Kacker, D. R. Kuhn, V. Okun, and J. Lawrence, "IPOG/IPOG-D: efficient test generation for multi-way combinatorial testing," *Software Testing, Verification and Reliability*, vol. 18, no. 3, pp. 125-148, 2008.
[38] A. W. Williams, "Determination of test configurations for pair-wise interaction coverage," *Testing of Communicating Systems: Tools and Techniques. IFIP TC6/WG6.1 13th International Conference on Testing of Communicating Systems (TestCom 2000)*, H. Ural, R. L. Probert and G. v. Bochmann, eds., pp. 59-74, Boston, MA: Springer US, 2000.
[39] B. S. Ahmed, K. Z. Zamli, and C. P. Lim, "Constructing a t-way interaction test suite using the particle swarm optimization approach," *International Journal of Innovative Computing, Information and Control*, vol. 8, no. 1, pp. 431-452, 2012.
[40] Y. Lei, and K.-C. Tai, "In-parameter-order: a test generation strategy for pairwise testing." pp. 254-261.
[41] M. I. Younis, and K. Z. Zamli, "MIPOG-an efficient t-way minimization strategy for combinatorial testing," *International Journal of Computer Theory and Engineering*, vol. 3, no. 3, pp. 388, 2011.
[42] D. M. Cohen, S. R. Dalal, M. L. Fredman, and G. C. Patton, "The AETG system: an approach to testing based on combinatorial design," *IEEE Transactions on Software Engineering*, vol. 23, no. 7, pp. 437-444, 1997.
[43] K. Z. Zamli, M. F. Klaib, M. I. Younis, N. A. M. Isa, and R. Abdullah, "Design and implementation of a t-way test data generation strategy with automated execution tool support," *Information Sciences*, vol. 181, no. 9, pp. 1741-1758, 2011.
[44] B. Jenkins. "Jenny tool," http://www.burtleburtle.net/bob/math.
[45] A. Williams. "TConfig tool," http://www.site.uottawa.ca/~awilliam.
[46] A. Hartman, T. Klinger, and L. Raskin, "IBM intelligent test case handler," *Discrete Mathematics*, vol. 284, no. 1, pp. 149-156, 2010.
[47] A. A. Hassan, S. Abdullah, K. Z. Zamli, and R. Razali, "Combinatorial Test Suites Generation Strategy Utilizing the Whale Optimization Algorithm," *IEEE Access*, vol. 8, pp. 192288-192303, 2020.
[48] K. Z. Zamli, B. Y. Alkazemi, and G. Kendall, "A tabu search hyper-heuristic strategy for t-way test suite generation," *Applied Soft Computing*, vol. 44, pp. 57-74, 2016.
[49] A. B. Nasser, and K. Z. Zamli, "A New Variable Strength t-way Strategy based on the Cuckoo Search Algorithm."
[50] A. B. Nasser, A. Alsewari, and K. Z. Zamli, "Learning cuckoo search strategy for t-way test generation." pp. 97-110.
[51] M. S. A. R. Ali, R. R. Othman, Z. R. Yahya, and M. Z. Zahir, "A Modified Artificial Bee Colony Based Test Suite Generation Strategy for Uniform T-Way Testing," *MS&E*, vol. 767, no. 1, pp. 012020, 2020.
[52] A. A. Alomoush, A. A. Alsewari, H. S. Alamri, K. Aloufi, and K. Z. Zamli, "Hybrid harmony search algorithm with grey wolf optimizer and modified opposition-based learning," *IEEE Access*, vol. 7, pp. 68764-68785, 2019.
[53] A. K. Alazzawi, H. M. Rais, and S. Basri, "Parameters tuning of hybrid artificial bee colony search based strategy for t-way testing," *Int. J. Innov. Technol. Exploring Eng.*, vol. 8, no. 5S, pp. 204-212, 2019.
[54] K. Z. Zamli, "An Improved Jaya Algorithm-Based Strategy for T-Way Test Suite Generation," *Emerging Trends in Intelligent Computing and Informatics: Data Science, Intelligent Information Systems and Smart Computing*, vol. 1073, pp. 352, 2019.
[55] H. Wang, Z. Wu, S. Rahnamayan, Y. Liu, and M. Ventresca, "Enhancing particle swarm optimization using generalized opposition-based learning," *Information Sciences*, vol. 181, no. 20, pp. 4699-4714, 2011.
[56] S. Rahnamayan, H. R. Tizhoosh, and M. M. Salama, "Quasi-oppositional differential evolution." pp. 2229-2236.
[57] M. Ergezer, D. Simon, and D. Du, "Oppositional biogeography-based optimization." pp. 1009-1014.
[58] Q. Xu, L. Wang, B. He, and N. Wang, "Opposition-based differential evolution using the current optimum for function optimization," *J. Appl. Sci*, vol. 29, no. 3, pp. 308-315, 2011.
[59] S. Rahnamayan, J. Jesuthasan, F. Bourennani, H. Salehinejad, and G. F. Naterer, "Computing opposition by involving entire population." pp. 1800-1807.
[60] R. Rao, "Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems," *International Journal of Industrial Engineering Computations*, vol. 7, no. 1, pp. 19-34, 2016.
[61] A. R. A. Alsewari, and K. Z. Zamli, "Design and Implementation of A Harmony-Search-Based Variable-Strength T-way Testing Strategy With Constraints Support," *Information and Software Technology*, vol. 54, no. 6, pp. 553-568, Jun, 2012.
[62] M. B. Cohen, "Designing test suites for software interaction testing," Doctoral dissertation, University of Auckland, 2004.
[63] Y. Lei, R. Kacker, D. R. Kuhn, V. Okun, and J. Lawrence, "IPOG: a general strategy for t-way software testing." pp. 549-556.
[64] T. Gordi, and H. Khamis, "Simple solution to a common statistical problem: interpreting multiple tests," *Clinical therapeutics*, vol. 26, no. 5, pp. 780-786, 2004.
[65] S. Holm, "A simple sequentially rejective multiple test procedure," *Scandinavian Journal of Statistics*, vol. 6, no. 2, pp. 65-70, 1979.

ABDULLAH B. NASSER received the B.Sc. degree from Hodeidah University, Yemen, in 2006, the M.Sc. degree from Universiti Sains Malaysia, Malaysia, in 2014, and the PhD degree in Computer Science from University Malaysia Pahang, in 2018, all in computer science. Currently, he is an Assistant Professor with Faculty of Computing, Universiti Malaysia Pahang, Malaysia. He is also the author of many scientific articles published in renowned journals and conferences. His research interests include Software Testing and Soft Computing, specifically, the use of artificial intelligence methods (meta-heuristic algorithms) for solving different software engineering problems.

KAMAL Z. ZAMLI received the degree in electrical engineering from the Worcester Polytechnic Institute, USA, in 1992, the M.Sc. degree in real-time software engineering from Universiti Teknologi Malaysia, in 2000, and the Ph.D. degree in software engineering from the University of Newcastle upon Tyne, U.K., in 2003. His main research area are (combinatorial t-way) software testing and search-based software engineering.

FADHL HUJAINAH is a post-doctoral fellow in Software Engineering at the joint Department of Computer Science and Engineering of Chalmers and University of Gothenburg in Sweden. Prior to that, he worked as a Senior Lecturer (Assistant Professor) at Faculty of Computing, University Malaysia Pahang in Malaysia. Fadhl received the B.Sc. degree (Hons.) in Software Engineering and the M.Sc. (Hons.) degree in Information Technology with first class honor grade from the Universiti Teknologi Malaysia, Malaysia, in 2012 and 2013, and Ph.D. degree in Software Engineering from Universiti Malaysia Pahang, Malaysia, in 2019. His research interests include software engineering with particular interest in requirements engineering, software architecture, stakeholder analysis, and decision making.

WAHEED ALI H. M. GHANEM received tthe B.Sc.degree in computer sciences and engineering from Aden University, Yemen, in 2003, and the M.Sc. degree in computer science and the Ph.D. degree in network and communication protocols from Universiti Sains Malaysia, in 2013 and 2019, respectively. His research interests include computer and network security, cybersecurity, machine learning, artificial intelligence, swarm intelligence, optimization algorithm, and information technology.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3055367, IEEE Access

ABDUL-MALIK H. Y. SAAD (Member, IEEE) was born in Jeddah, Saudi Arabia, in 1983. He received the B.Eng. degree with the first rank in computer engineering from Hodeidah University, Hodeidah, Yemen, in 2006, the M.Sc. degree in electronic systems design engineering from Universiti Sains Malaysia (USM), in 2014, and the Ph.D. degree in the digital systems field from USM, in 2018. He is currently an Assistant Professor with School of Electrical Engineering, Faculty of Engineering, Universiti Teknologi Malaysia. His research interests include artificial intelligent (AI), digital system design, and image processing.

N.A.M. ALDUAIS: "Nayef Abdulwahab Mohammed Alduais received the B.Eng. degree with first rank in Computer Engineering from Hodeidah University,Yemen, in 2007, and the Master and Ph.D. degree in Communication and Computer Engineering from Faculty of Electrical and Electronic Engineering(FKEE), Universiti Tun Hussein Onn Malaysia (UTHM), Malaysia, in 2015 and 2019, respectively. He is currently a Lecturer and researcher in the Internet of Things (IoT) at Faculty of Computer Science and Information Technology (FSKTM), UTHM, having previously worked as an Assistant Lecturer with the Faculty of Computer Science and Engineering, Hodeidah University, Yemen, from 2007 to 2013. He has authored numerous papers in journals and conference proceedings. His research interests include WSN , IoT, Edge computing , and Artificial intelligence(AI). He received numerous medals and scientific excellence certificates.